International Journal of Electronics Engineering, 1(2), 2009, pp. 265-268

Design and Implementation of Frequency Analyzer Using VHDL

Anant G. Kulkarni[1] & Sudha Nair[2]

Department of Electronics & Telecommunication, R.C.E.T, CSVTU University, Bhilai,(C.G), INDIA

Abstract: *In this paper, we propose a VHDL based design of a frequency analyzer that implements a Fast Fourier. The module had been developed using radix-2 decimation in time algorithm of n-point samples. Structural and Behavioral modeling was implemented using VHDL to describe, simulate, and perform the design. The resulting design was simulated using Xilinx ISE 9.1i and Modelsim SE 6.3f and can be synthesized on Spartan 3E DSP development board. The simulation results are presented in this paper.*

Keywords*: FFT, Architecture, Butterfly unit, Simulation, Xilinx, Modelsim.*

1. INTRODUCTION

Nowadays semiconductor technology is able to create very complex devices that can enclose a complete system in a single chip (SoC). If the system is created from scratch, achieving the desired performance is costly and time consuming. To meet the tight time-to-market requirement, the electronic design uses pre-designed intellectual property (IP) cores as a common practice. These cores may be parametrizable and customizable to be synthesized in a large application specification. They are available to the designer from heterogeneous sources, design team, CAD tool libraries, CAD tool independent libraries, etc.

One of the areas that major demands of application specific circuits design is digital signal processing (DSP). Fast Fourier Transform is a computationally intensive DSP function, widely used in many applications. Since the pioneering work by Cooley and Tukey (Cooley and Tukey, 1965), a lot of work has been done on the FFT algorithm such as the radix-2^m algorithm and the split radix algorithm (Brigham, 1998; Winograd, 1976; Duhamel and Hallmann, 1984). Among them, the radix-2 and radix-4 algorithms have been used mostly for practical applications due to their simple structures.Most of the implementations and benchmarking of FFT algorithms has been done using general purpose processors, DSP processors and dedicated FFT.This paper provides a novel approach to the design of frequency analyzer using FFT algorithm.

2. FAST FOURIER TRANSFORM

The N-point Discrete Fourier Transform (DFT) of a finite duration sequence $x(n)$ is defined as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk} k = 0,1,...N-1 \quad (1)$$

where $W = e^{-j(2\pi/N)}$ is referred as the twiddle factor, N is the transform size and $j = \sqrt{-1}$.

The FFT is an efficient algorithm to compute the DFT and its inverse (Cooley and Tukey, 1965; Brigham, 1998). It generally falls into two classes: Decimation In Time (DIT), and Decimation In Frequency (DIF). The DIT algorithm first rearranges the input elements in bit-reversed order and then builds the output transform. The DIF algorithm first transforms and then rearranges the output values. The basic idea of these algorithms is to break up an N-point DFT transform into successive smaller and smaller transform known as a butterfly (basic computational element). The smallest transform used is a 2-point DFT known as radix-2, it processes groups of 2 samples. The combination of two stages of radix-2 in one stage constitute radix-4 algorithms, it processes groups of 4 samples. There are also other decomposition schemes known as split-radix algorithms (Duhamel and Hallmann, 1984).

The calculations implied in the basic computational element (butterfly) for radix-2 in DIT algorithm will be introduced next. A and B are two complex numbers represented as:

$$A = x + jX \quad (2)$$

$$B = y + jY \quad (3)$$

where x and y are real parts and X and Y are imaginary of them. "A transform (A')" and "B transform (B')" are calculated as shown the next equations:

$$A' = x' + jX' = A + B\, W_N^k \quad (4)$$

$$B' = y' + jY' = A - B\, W_N^k \quad (5)$$

*Corresponding Author: agk30@rediffmail.com

$$W_N^k = \cos(2\pi\, k/N) - j\, \sin(2\pi\, k/N) \quad (6)$$

Taking into consideration (2), (3) and (6), the transforms may be written as:

$$A' = [(x + y\cos(2\pi\, k/N) + Y\sin(2\pi\, k/N)) + j(X + Y\cos(2\pi\, k/N) - y\sin(2\pi\, k/N))] \quad (7)$$

$$B' = [(x - y\cos(2\pi\, k/N) - Y\sin(2\pi\, k/N)) + j(X - Y\cos(2\pi\, k/N) + y\sin(2\pi\, k/N))] \quad (8)$$

where k depends of the number of stages and the number of samples.

3. VHDL MODELING

The objective of this paper is to implement in an efficient manner the equations (7) and (8) having in mind the reusability of the resulting design as embedded core in a possible wide range of applications. Then the number of samples N and the number of bits to coding each sample must be considered as generic parameters in order to adapt the size to the specific application and to control the quantization errors when using fixed point arithmetic. Also the degradation performance should be considered with the increment of the number of samples N. All components of the hardware used in this paper have been modeled in VHDL according to the restrictions and recommendations for high level behavioral synthesis (Keating and Bricand, 2002).

4. ARCHITECTURAL DESIGN OF ANALYZER

The operation of the analyzer is partitioned into three main processes. These are the Data Input, FFT Computation and Data Output Processes. This partitioning is depicted in Figure 1.

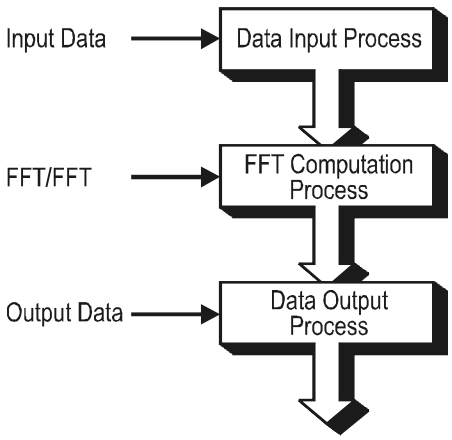

Figure 1: Operation of the Analyzer

The processing cycle starts with the Data input process, during which sampled data is read in and stored in memory. During the FFT computation process, the FFT is computed on the stored data. During the Output process results of the FFT computation process are read out to the outside world. These processes are then mapped to hardware resources.

4.1. Block Diagram of the Analyzer

The FFT architecture consists of a single radix-2 butterfly (which is referred as the butterfly processing element), a dual-port FIFO RAM, a coefficient ROM, a controller and an address generation unit. It also consists of a "cycles unit" to separate the various cycles, namely c0, c1, c2 and c3. This unit also outputs the ORed output of some of these cycles such as c0 and c1. The process of writing into the RAM during FFT computation begins only five cycles after the first data is read from RAM. The counter unit is used to count these cycles. Data pathways are in the form of 32-bit signed fractions. Coefficients are stored as 32-bit words.

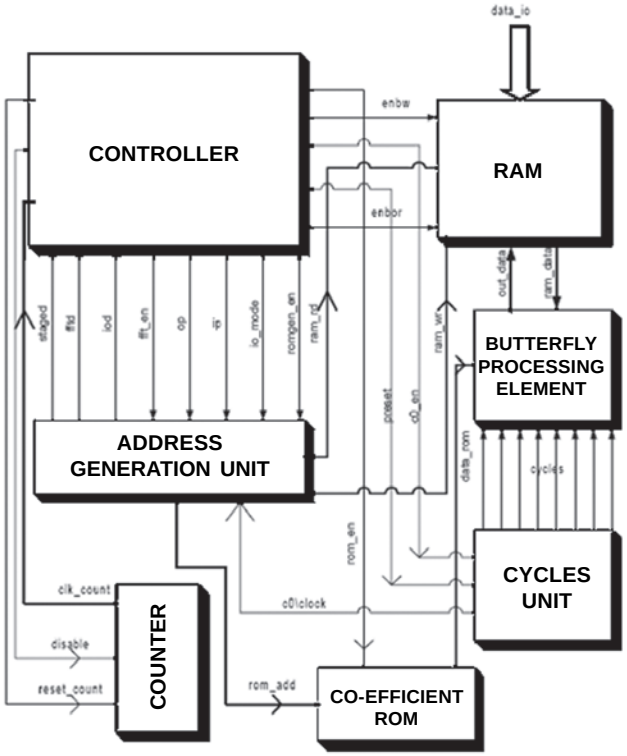

Figure 2: The FFT Processor

4.2. Butterfly Processing Element

The butterfly is the basic operator of the FFT. It computes a two point FFT. It takes two data words from memory and computes the FFT. The results are written back to the same memory locations of the inputs since an in-place algorithm is used. The butterfly processing element computes one butterfly every four cycles. It consists of one multiplier and two adders. The architecture for it is depicted in Figure 3. The blocks named "R" are a set of negative edge triggered D flip flops. That is, each "R" block consists of 32 D-flip flops, one for each bit. Similarly the "L" blocks are positive level triggered. The blocks labeled "D" are positive edge triggered. c0, c1, c2 and c3 are the four cycles that the processor takes to calculate the fft. c0, c1 is the OR output of the cycles c0 and c1. The multiplier forms the partial

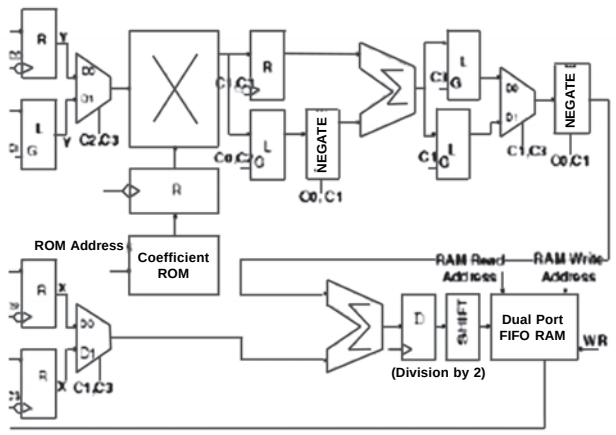

Figure 3: Butterfly Processing Element

products of the complex multiplication and produces a 32 bit signed fraction result. This is followed by the first adder which sums the cross products of the complex multiplication. The second adder produces the sum and difference outputs of the butterfly operation. The butterfly processing element takes four cycles to compute a two-point FFT. It has a latency of five cycles. Three of these are associated with the fact that three input components (*y*, *Y*, and *x*) are required before an output can be computed and two are to pipeline the RAM read and write operations.

4.3. Address Generation Unit (AGU)

The purpose of the address generation unit is to provide the RAM and the coefficient ROM with the correct addresses. It also keeps track of which butterfly is being computed in which stage. For an 8-point complex FFT there are 3 stages, each stage consisting of 4 butterflies. In addition to this, since address generation during input, output and FFT computation processes are different; it keeps track of the mode of operation of the chip and generates the required address. Mode of operation information is supplied by the controller. The different blocks of the AGU are explained separately.

4.3.1. Butterfly Generator

The butterfly generator keeps track of which butterfly is being computed in a particular stage. It is basically a 16-bit up counter since for an 8 point complex FFT there are 4 butterflies per stage and 4 data words per butterfly (2 real and 2 imaginary).Note that during data input and data output the butterfly is incremented by the clock while during fft computation mode, it is incremented by c0. This is because, 4 cycles are required to calculate one butterfly. Hence the butterfly generator needs to be incremented only once in every 4 cycles during FFT computation. The selection between the clock and "c0" is made by a multiplexer. The "io mode" signal is used for selection. Whenever "clear" or "stage done" signal goes high, the butterfly generator is reset.

4.3.2. Stage Generator

The stage generator keeps track of the current stage in the FFT computation. The stage generator supplies the base index generator with the number of the stage which is currently being computed. For an 8 point FFT there are 3 stages hence the stage generator is basically a two bit counter which is incremented one every 4 butterfly counts (by the "stage done" signal).

4.3.3. Stage Done_IO Done Block

It generates four signals called "iod", "staged" "fftd" and "butterfly". "iod" is generated when the "butterfly" count is 15. This informs the controller that either the Data Input or Output process is finished. The "staged" signal is generated when the current "butterfly" count is 4, it increments the stage generator by one. fftd is generated when the stage count is three. This informs the controller that the FFT computation process is done, hence forcing the FFT processor to start the data output process.

4.3.4. IO-Address Generator

The IO Address Generator is responsible for generating addresses for RAM during the data input and output processes. During the data input process the output of the butterfly generator "butterfly" can be used for addressing 16 locations in the RAM. However, during the data output process data should be bit-reversed while being written to outside world. Once in the output process bit-reversed address is selected by the muxes in the AGU. The controller gives the information whether the process is in IO-mode through the signal "iomode". This signal is used for selecting. The butterfly has two complex data inputs A and B. These inputs when manipulated produces four outputs *x*, *X*, *y* and *Y*, out of which X and Y are complex values. Since there are 16 locations, the BIG is a mode-16 counter. The FFT mode address generation is quite complex. The address generation is obtained by manipulating the outputs of the butterfly generator, stage generator and the cycles.

4.3.6. The Shifters

As mentioned, the result of FFT computation is written back into the same location as it was read. However there is a latency of five cycles. For example, if "y" is read from the RAM during cycle "c0", "y^1" is written into the same location as it was read after 5 cycles that is during cycle "c1". So the read address is shifted in each on these five cycles. The output of the last shifter is then given as the write address.

4.3.7. ROM Address Generator

The ROM Address Generator is used to provide the ROM with the correct address for collecting the sine and cosine co-efficients. It is modeled based on the co-efficients given in the signal flow graph.

4.4. Controller

The controller is modeled as a finite state machine which has been explained already. It has seven states ranging from rst1 to rst 7. The actions performed in each state are clearly commented in the code.

4.5 RAM and ROM

The input is first written into the RAM. During the FFT computation process, the FFT of two numbers is calculated and written back into the same location in the RAM. During the output process bit reversed address is given to the RAM and it outputs the data in it accordingly. The ROM is used to store the sine and cosine values needed in the FFT computation process. It outputs these values according to the address given to it.

5. RESULTS AND DISCUSSION

RTL Simulation of the Frequency Analyzer

The frequency analyzer unit was simulated and synthesized separately using VHDL with the help of Xilinx ISE 9.1i and Model Sim SE 6.3f. The butterfly unit, the address generation unit, the controller and the ROM units were simulated and synthesized separately. The results shown here include the RTL and simulation of the top analyzer unit.

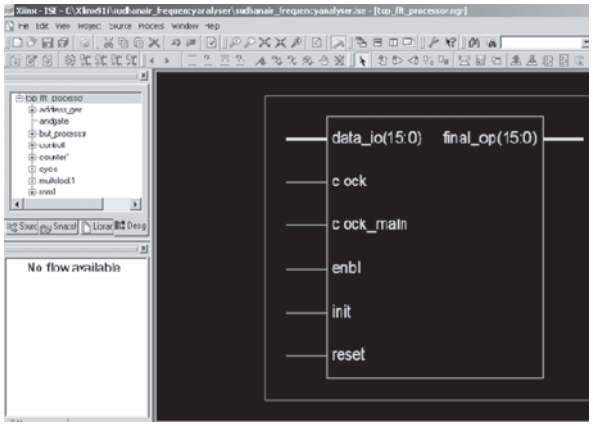

Figure 4: RTL of the Frequency Analyzer.

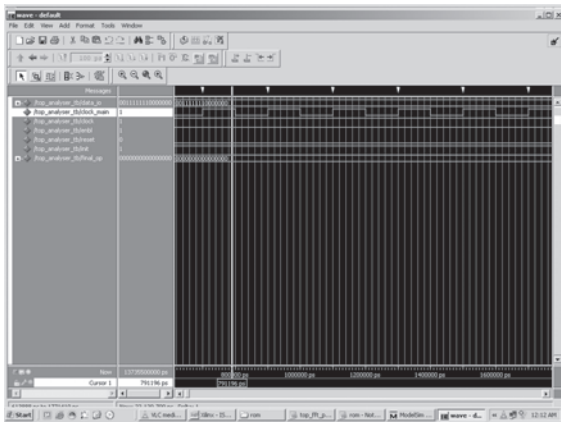

Figure 5: Simulation result of the Analyzer.

6. CONCLUSION

In this paper a novel approach for frequency analyzer has been presented. The modeling, simulation and analysis of the design are presented. The frequency analyzer unit along with its components was separately simulated and synthesized using Xilinx ISE 9.1i and ModelSim 6.3f.A test bench is written for the analyzer which reads different inputs from a file. The results obtained were seen to be matched with that obtained using Mat lab. An 8 point FFT processor was designed, simulated and synthesized using VHDL. The chip can be easily upgraded for a 128 point or 256 point FFT. The complete design can be implemented on Spartan 3E DSP development board.

REFERENCES

[1] Cooley, J. W., and J. W. Tukey, "An Algorithm for the Machine Calculation of the Complex Fourierseries", *Math. of Computation*, **19,** (1965), 297-301.

[2] Brigham, E. O., The Fast Fourier Transform and its Applications, Prentice Hall, (1998).

[3] Winograd S., "On Computing the Discrete Fourier Transform", Proc. Nat. Acad. Sci. U.S., **73,** (1976), 1005-1006.

[4] Duhamel, P., and H. Hallmann, "Split Radix FFT Algorithm", *IEEE Electronic Letters*, **20**, (1984), 14-16.

[5] Keating, M. and P. Bricand, "Reuse Methodology Manual: for System-on-a-Chip Designs. Third Edition, Kluwer Academic Publishers (2002).

[6] VHDL Modeling and Model Testing for DSP Applications Armstrong, J. R. Gray, F. G. Meng-Wei Lin Bradley, Dept. of Electr. & Comput. Eng., Virginia Polytech. Inst. & State Univ., Blacksburg, VA; Industrial Electronics, *IEEE Transactions*, **46**, (1999).

[7] Mauricio Ayala-Rinc?, Rodrigo B. Nogueira, Carlos H. Llanos, Ricardo P. Jacobi, Reiner W. Hartenstein, "Modeling a Reconfigurable System for Computing the FFT in Place via Rewriting-Logic," sbcci, 16th Symposium on Integrated Circuits and Systems Design (SBCCI'03), (2003), 205.

[8] Syddyka Berna Örs, Ahmet Dervisoglu, "Modeling Bit Multiplication Blocks for DSP Applications Using VHDL," Euromicro, 25th Euromicro Conference (EUROMICRO '99), **1,** (1999), 1402.

[9] J. Andersson, "A DSP ASIC Design Flow Based on VHDL and ASIC-emulation," Eurodac, European Design Automation Conference with EURO-VHDL, (1995), 562.

[10] Synthesis Modeling Techniques in VHDL, *Journal of Engineering Technology*, Fall 2002 by Nowlin, Robert W., Soman, Sanjiv C., Sundararajan, Raji D.